CFD Validation and Adaptivity for Viscous Flow Simulations

Victorien Menier, Adrien Loseille, Frédéric Alauzet

CFD Validation and Adaptivity for Viscous Flow Simulations

Victorien Menier*, Adrien Loseille† and Frédéric Alauzet‡
Gamma3 Team, INRIA Paris-Rocquencourt, 78153 Le Chesnay, France

This article deals with the description and the validation of the unstructured viscous RANS flow solver `Wolf` and its compliance with anisotropic unstructured mesh adaptation. We first describe the numerical scheme of the flow solver and its validation on standard test cases. We then give a general description of the mesh generator and its main features for viscous mesh generation and adaptation. From these points, we define some simple strategies to perform adaptive computation for RANS solutions. Each technique is illustrated with a numerical example and we discuss its robustness, effectiveness and ability to handle complex geometries.

Introduction

Mesh adaptation provides a way to control the accuracy of the numerical solution by modifying the domain discretization according to size and directional constraints. When dealing with real life flow problems, Hessian-based unstructured mesh adaptation has already proved its efficiency to improve the ratio between the solution accuracy and the number of degrees of freedom (the problem complexity).[1,2,3,4,5,6] In addition, as a large number of physical phenomena are anisotropic by nature, anisotropic mesh adaptation improves even more this ratio.

In the context of high Reynolds compressible viscous flow simulations though, mesh adaptation remains challenging. It is notably due to steep gradients in the near-wall regions (boundary layers). Dramatic variations in the normal direction of variables such as the velocity require specific meshing techniques. The generation of quasi-structured meshes in the near wall regions has proved to be a reliable approach but their integration in the mesh adaptation loop is difficult. Indeed, to perform a mesh adaptation for a viscous flow simulation, one needs a reliable flow solver and suitable meshing technologies. The scope of this paper is to apply our flow solver and meshing technologies to adaptive simulations of viscous flows.

The paper is organized as follows. In Section I, we describe the `Wolf` flow solver. We focus on the CFD modeling, the turbulence model used, time step computation, numerical fluxes computation. Code verification and validation is based on comparison with other codes on standard test cases. In Section II, we describe the local remeshing strategies. In particular, we describe the algorithms involved for surface, volume, boundary layer and anisotropic cartesian mesh generation in an adaptive context. In Section III, we introduce four adaptive approaches for viscous simulations: (i) a fully unstructured, (ii) an hybrid with anisotropic element and a frozen boundary layer, (iii) an hybrid with anisotropic element and a regenerated boundary layer, and (iv) a fully adaptive cartesian approach.

I. Implementation and validation of the flow solver `Wolf`

This section details the in-house compressible flow solver `Wolf` to model viscous flows. We first describe a few key characteristics of its implementation: CFD modeling, turbulence model used, finite element/finite volume spatial discretization and the time implicit discretization. The intent of this section is not to describe the whole implementation in details but to give its main characteristics. Finally, we exhibit some results of the code verification and validation study that we led.

*PhD student, Gamma3 Team, recipient of a PHD grant from the Airbus Group Foundation
†Researcher, Gamma3 Team
‡Researcher, Gamma3 Team

A. Modeling equations

The Reynolds Averaged Navier-Stokes (RANS) system relying to the Spalart-Allmaras model is composed of the compressible Navier-Stokes equations and the standard Spalart-Allmaras equation with no trip.

THE COMPRESSIBLE NAVIER-STOKES EQUATIONS. The compressible Navier-Stokes equations for mass, momentum and energy conservation reads:

$$\left\{\begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) & = & 0\,, \\ \dfrac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p & = & \nabla \cdot (\mu \mathcal{T})\,, \\ \dfrac{\partial (\rho e)}{\partial t} + \nabla \cdot ((\rho e + p)\mathbf{u}) & = & \nabla \cdot (\mu \mathcal{T} \mathbf{u}) + \nabla \cdot (\lambda \nabla T)\,, \end{array}\right.$$

where ρ denotes the density, $\mathbf{u}$ the velocity, e the total energy per mass, p the pressure, T the temperature, μ the laminar dynamic viscosity and λ the laminar conductivity. $\mathcal{T}$ is the laminar stress tensor:

$$\mathcal{T} = (\nabla \otimes \mathbf{u} + {}^t\nabla \otimes \mathbf{u}) - \frac{2}{3} \nabla . \mathbf{u}\, \mathbb{I}\,.$$

The variation of nondimensionalized laminar dynamic viscosity and conductivity coefficients μ and λ as a function of a dimensional temperature T is defined by the Sutherland law.

$$\mu \;=\; \mu_\infty \left(\frac{T}{T_\infty}\right)^{\frac{3}{2}} \left(\frac{T_\infty + \mathrm{Su}}{T + \mathrm{Su}}\right) \quad \text{and} \quad \lambda \;=\; \lambda_\infty \left(\frac{T}{T_\infty}\right)^{\frac{3}{2}} \left(\frac{T_\infty + \mathrm{Su}}{T + \mathrm{Su}}\right)\,,$$

where $\mathrm{Su} = 110$ is the Sutherland constant and the index ∞ denotes reference quantities. The relation linking μ and λ is expressed from the Prandtl laminar number:

$$\mathrm{Pr} = \frac{\mu c_p}{\lambda} \quad \text{with} \quad \mathrm{Pr} = 0.72 \quad \text{for (dry) air}\,.$$

This system can be rewritten under vectorial form:

$$W_t + F_1(W)_x + F_2(W)_y + F_3(W)_z = S_1(W)_x + S_2(W)_y + S_3(W)_z$$

where W is the nondimensionalized conservative variables vector:

$$W = (\rho, \rho u, \rho v, \rho w, \rho E)^T$$

$\mathbf{F}(W) = (F_1(W), F_2(W), F_3(W))$ are the convective (Euler) flux functions:

$$\begin{array}{rcl} F_1(W) & = & (\rho u,\ \rho u^2 + p,\ \rho uv,\ \rho uw,\ u(\rho E + p))^T \\ F_2(W) & = & (\rho v,\ \rho uv,\ \rho v^2 + p,\ \rho vw,\ v(\rho E + p))^T \\ F_3(W) & = & (\rho w,\ \rho uw,\ \rho vw,\ \rho w^2 + p,\ w(\rho E + p))^T \end{array}$$

and $\mathbf{S}(W) = (S_1(W), S_2(W), S_3(W))$ the laminar viscous fluxes:

$$\begin{array}{rcl} S_1(W) & = & (0,\ \tau_{xx},\ \tau_{xy},\ \tau_{xz},\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \lambda \mathcal{T}_x)^T \\ S_2(W) & = & (0,\ \tau_{xy},\ \tau_{yy},\ \tau_{yz},\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} + \lambda \mathcal{T}_y)^T \\ S_3(W) & = & (0,\ \tau_{xz},\ \tau_{yz},\ \tau_{zz},\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + \lambda \mathcal{T}_z)^T \end{array}$$

where τ_{ij} are the components of laminar stress tensor defined by:

$$\tau_{ij} = \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - \frac{2}{3} \mu \frac{\partial v_k}{\partial x_k} \delta_{ij}\,.$$

where (v_i, v_j, v_k) are the three components of the velocity and δ_{ij} the Kroneker symbol.

TURBULENCE MODELING. The chosen turbulence model is the one equation Spalart-Allmaras model:[7]

$$\frac{\partial \tilde{\nu}}{\partial t} + \mathbf{u}\cdot\nabla\tilde{\nu} = c_{b1}[1-f_{t2}]\tilde{S}\tilde{\nu} - \left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{1}{\sigma}\left[\nabla\cdot((\nu+\tilde{\nu})\nabla\tilde{\nu}) + c_{b2}\|\nabla\tilde{\nu}\|^2\right] + f_{t1}\Delta\mathbf{u}^2\,.$$

where $\tilde{\nu}$ is the turbulent kinematic viscosity. In the standard model the trip term is being left out, *i.e.*, $f_{t1}=0$. Moreover, some implementations ignore also the f_{t2} term as it is argued that if the trip is not included, then f_{t2} is not necessary.[8] In `Wolf`, this simplified version has been considered and we prefered to write it under the following form which is more appropriate for its discretization with the finite element/finite volume method:

$$\frac{\partial \rho\tilde{\nu}}{\partial t} + \underbrace{\mathbf{u}\cdot\nabla\rho\tilde{\nu}}_{convection} = \underbrace{c_{b1}\tilde{S}\rho\tilde{\nu}}_{production} \underbrace{- c_{w1}f_w\rho\left(\frac{\tilde{\nu}}{d}\right)^2}_{destruction} + \underbrace{\frac{\rho}{\sigma}\nabla\cdot((\nu+\tilde{\nu})\nabla\tilde{\nu})}_{dissipation} + \underbrace{\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2}_{diffusion}$$

Notice that this is not a conservative model. If conservative form of the Spalart-Allmaras is foreseen, we have to consider the variation of Catris and Aupoix.[9] The turbulent eddy viscosity is computed from:

$$\mu_t = \rho\tilde{\nu}f_{v1}$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3+c_{v1}^3} \quad \text{and} \quad \chi = \frac{\tilde{\nu}}{\nu} \quad \text{with} \quad \nu = \frac{\mu}{\rho}\,.$$

Additional definitions are given by the following equations:

$$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}} \quad \text{and} \quad \tilde{S} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2}f_{v2} \quad \text{where} \quad \Omega = \|\nabla\times\mathbf{u}\|\,.$$

with d the distance to nearest wall which is computed for each vertex at the beginning of the simulation. The constants are

$$\sigma = \frac{2}{3} \qquad c_{b1}=0.1355 \quad c_{b2}=0.622 \quad \kappa = 0.41$$

$$c_{w1} = \frac{c_{b1}}{\kappa} + \frac{1+c_{b2}}{\sigma} \quad c_{w2}=0.3 \quad c_{w3}=2 \quad c_{v1}=7.1\,.$$

Finally, the function f_w is computed as:

$$f_w = g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{1/6} \quad \text{with} \quad g = r + c_{w2}\left(r^6-r\right) \quad \text{and} \quad r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}\,,\,10\right)\,.$$

B. Spatial discretization

The spatial discretization of the fluid equations is based on a vertex-centered finite element/finite volume formulation on unstructured meshes. It combines a HLLC upwind schemes for computing the convective fluxes and the Galerkin centered method for evaluating the viscous terms. Second order space accuracy is achieved through a piecewise linear interpolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure which uses a particular edge-based formulation with upwind elements. A specific slope limiter is employed to damp or eliminate spurious oscillations that may occur in the vicinity of discontinuities.

1. *Finite Volume discretization*

Let $\mathcal{H}$ a mesh of domain Ω, the vertex-centered finite volume formulation consists in associating with each vertex P_i of the mesh a control volume or finite volume cell, denoted C_i. Discretized domain Ω_h can be written as the union of the elements or the union of the finite volume cells:

$$\Omega_h = \bigcup_{i=1}^{N_T} K_i = \bigcup_{i=1}^{N_S} C_i\,.$$

In order to discretize accurately the flow equations on highly anisotropic meshes, the dual finite volume cells are the containment sphere cells introduced in 2D by Barth[10] and generalized to 3D by Dervieux[11] instead of the classic median cells. It consists in subdividing each tetrahedron into four hexahedra cell around each vertex. The hexahedron cell vertices associated with vertex P_i are (i) the middle of the three edges issued from P_i, (ii) the containment circle center of the three faces containing P_i, (iii) the containment sphere center of the tetrahedron and (iv) the considered vertex P_i. The containment sphere cells of vertex P_i is the union of all its hexahedra cells. The containment sphere center correspond the the sphere circumcenter if it falls inside the element.

The Reynolds Averaged Navier-Stokes equations integrated on the finite volume cell C_i following a finite volume formulation reads (using the Green formula):

$$|C_i|\frac{dW_i}{dt} + \int_{\partial C_i} \mathbf{F}(W_i)\cdot\mathbf{n}_i \,\mathrm{d}\gamma = \int_{\partial C_i} \mathbf{S}(W_i)\cdot\mathbf{n}_i \,\mathrm{d}\gamma\,, + \int_{C_i} \mathbf{Q}(W_i)\,\mathrm{d}\mathbf{x} + \text{B.T.} \tag{1}$$

where W_i is the mean value of the solution W on the cell C_i, $\mathbf{n}_i$ is the outer normal to the finite volume cell surface ∂C_i, $\mathbf{F}, \mathbf{S}$ and $\mathbf{Q}$ are respectively the convective, viscous and source terms and B.T. represents boundary conditions terms.

2. *Discretization of the convective terms*

The integration of convective fluxes $\mathbf{F}$ of Equation (1) is done by decomposing the cell boundary in many facets ∂C_{ij}:

$$\int_{\partial C_i} \mathbf{F}(W_i)\cdot\mathbf{n}_i \,\mathrm{d}\gamma \;=\; \sum_{P_j\in\mathcal{V}(P_i)} \mathbf{F}|_{\partial C_{ij}} \cdot \int_{\partial C_{ij}} \mathbf{n}_i \,\mathrm{d}\gamma\,,$$

where $\mathcal{V}(P_i)$ is the set of all neighboring vertices linked by an edge to P_i and $\mathbf{F}|_{\partial C_{ij}}$ represents the constant value of $\mathbf{F}(W)$ at interface ∂C_{ij}. The flow is calculated with a numerical flux function, denoted Φ_{ij}:

$$\Phi_{ij} = \Phi_{ij}(W_i, W_j, \mathbf{n}_{ij}) = \mathbf{F}|_{\partial C_{ij}} \cdot \int_{\partial C_{ij}} \mathbf{n}_i \,\mathrm{d}\gamma\,,$$

where $\mathbf{n}_{ij} = \displaystyle\int_{\partial C_{ij}} \mathbf{n}_i \,\mathrm{d}\gamma$. The numerical flux function approximates the hyperbolic terms on the common boundary ∂C_{ij}. We notice that the computation of the convective fluxes is performed mono-dimensionally in the direction normal to the boundary of the finite volume cell. Therefore, the numerical calculation of the flux function Φ_{ij} at the interface ∂C_{ij} is achieved by the resolution of a one-dimensional Riemann problem in the direction of the normal $\mathbf{n}_{ij}$ by means of an approximate Riemann solver. In this work, the HLLC approximate Riemann solver is used for the mean flow - more details can be found[12] - and linear upwind advection is used for the turbulent variable convection.

HLLC approximate Riemann solver. The idea of the HLLC flow solver is to consider locally a simplified Riemann problem with two intermediate states depending on the local left and right states. The simplified solution to the Riemann problem consists of a contact wave with a velocity S_M and two acoustic waves, which may be either shocks or expansion fans. The acoustic waves have the smallest and the largest velocities (S_L and S_R, respectively) of all the waves present in the exact solution. If $S_L > 0$ then the flow is supersonic from left to right and the upwind flux is simply defined from $\mathbf{F}(W_l)$ where W_l is the state to the left of the discontinuity. Similarly, if $S_R < 0$ then the flow is supersonic from right to left and the flux is defined from $\mathbf{F}(W_r)$ where W_r is the state to the right of the discontinuity. In the more difficult subsonic case when $S_L < 0 < S_R$ we have to calculate $\mathbf{F}(W_l^*)$ or $\mathbf{F}(W_r^*)$. Consequently, the HLLC flux is given by:

$$\Phi_{lr}^{hllc}(W_l, W_r, \mathbf{n}_{lr}) = \begin{cases} \mathbf{F}(W_l)\cdot\mathbf{n}_{lr} & \text{if } S_L > 0 \\ \mathbf{F}(W_l^*)\cdot\mathbf{n}_{lr} & \text{if } S_L \le 0 < S_M \\ \mathbf{F}(W_r^*)\cdot\mathbf{n}_{lr} & \text{if } S_M \le 0 \le S_R \\ \mathbf{F}(W_r)\cdot\mathbf{n}_{lr} & \text{if } S_R < 0 \end{cases}$$

Now, let us specify how W_l^* and W_r^* are evaluated. We denote by $\eta = \mathbf{u}\cdot\mathbf{n}$. Assuming that $\eta^* = \eta_l^* = \eta_r^* = S_M$, the following evaluations are proposed[12] (the subscript $_l$ or $_r$ are omitted for clarity):

$$W^* = \frac{1}{S-S_M}\begin{pmatrix} \rho\,(S-\eta) \\ \rho\mathbf{u}\,(S-\eta) + (p^*-p)\mathbf{n} \\ \rho E\,(S-\eta) + p^* S_M - p\eta \end{pmatrix} \quad \text{where} \quad p^* = \rho\,(S-\eta)(S_M-\eta) + p\,.$$

A key feature of this solver is in the definition of the three waves velocity. For the contact wave we consider:

$$S_M = \frac{\rho_r\eta_r(S_R-\eta_r) - \rho_l\eta_l(S_L-\eta_l) + p_l - p_r}{\rho_r(S_R-\eta_r) - \rho_l(S_L-\eta_l)}\,,$$

and the acoustic wave speeds based on Roe average:

$$S_L = \min(\eta_l - c_l, \tilde{\eta} - \tilde{c}) \quad \text{and} \quad S_R = \max(\eta_r + c_r, \tilde{\eta} + \tilde{c})\,.$$

With such waves velocities, the HLLC Riemann solver has the following properties. It automatically (i) satisfies the entropy inequality, (ii) resolves isolated contacts exactly, (iii) resolves isolated shocks exactly, (iv) preserves positivity.

Linear convection. The turbulent variable $\tilde{\nu}$ is linearly convected:

$$\Phi_{ij}^{\rho\tilde{\nu}}(W_i, W_j, \mathbf{n}_{ij}) = \begin{cases} \eta\,\rho\tilde{\nu}_i & \text{if} \quad \eta > 0 \\ \eta\,\rho\tilde{\nu}_j & \text{otherwise} \end{cases} \qquad \text{where} \quad \eta = \frac{1}{2}\left(\mathbf{u}_i\cdot\mathbf{n}_{ij} + \mathbf{u}_j\cdot\mathbf{n}_{ij}\right)\,.$$

High-order accurate version. The MUSCL type reconstruction method has been designed to increase the order of accuracy of the scheme.[13] The idea is to use extrapolated values W_{ij} and W_{ji} of W at the interface ∂C_{ij} to evaluate the flux. The following approximation is performed:

$$\Phi_{ij} = \Phi_{ij}(W_{ij}, W_{ji}, \mathbf{n}_{ij})\,,$$

with W_{ij} and W_{ji} which are linearly interpolated as:

$$W_{ij} \;=\; W_i + \frac{1}{2}\,(\nabla W)_{ij}\cdot\overrightarrow{P_iP_j} \qquad \text{and} \qquad W_{ji} \;=\; W_j + \frac{1}{2}\,(\nabla W)_{ji}\cdot\overrightarrow{P_jP_i}\,,$$

where, in contrast to the original MUSCL approach, the approximate "slopes" $(\nabla W)_{ij}$ and $(\nabla W)_{ji}$ are defined for any edge and obtained using a combination of centered, upwind and nodal gradients.

The centered gradient, which is related to edge P_iP_j, is defined as:

$$(\nabla W)_{ij}^C\cdot\overrightarrow{P_iP_j} = W_j - W_i\,.$$

Upwind and downwind gradients, which are also related to edge P_iP_j, are computed according to the definition of upstream and downstream tetrahedra of edge P_iP_j. These tetrahedra are respectively denoted K_{ij} and K_{ji}. K_{ij} (resp. K_{ji}) is the unique tetrahedron of the ball of P_i (resp. P_j) the opposite face of which is crossed by the line defined by the edge P_iP_j. Upwind and downwind gradients are then defined for vertices P_i and P_j as:

$$(\nabla W)_{ij}^U = (\nabla W)|_{K_{ij}} \quad \text{and} \quad (\nabla W)_{ij}^D = (\nabla W)|_{K_{ji}}\,.$$

where $(\nabla W)|_K = \sum_{P\in K} W_P\nabla\phi_P|_K$ is the P_1-Galerkin gradient on tetrahedron K. Parametrized nodal gradients are built by introducing the β-scheme:

$$\begin{aligned} (\nabla W)_{ij} &= (1-\beta)(\nabla W)_{ij}^C + \beta\,(\nabla W)_{ij}^U \\ (\nabla W)_{ji} &= (1-\beta)(\nabla W)_{ij}^C + \beta\,(\nabla W)_{ij}^D\,, \end{aligned}$$

where $\beta\in[0,1]$ is a parameter controlling the amount of upwinding. For instance, the scheme is centered for $\beta = 0$ and fully upwind for $\beta = 1$.

NUMERICAL DISSIPATION OF FOURTH-ORDER: V4-SCHEME. The most accurate β-scheme is obtained for $\beta = 1/3$. Indeed, it can be demonstrated that this scheme is third-order for the two-dimensional linear advection on structured triangular meshes. On unstructured meshes, a second-order scheme with a fourth-order numerical dissipation is obtained. These high-order gradients are given by:

$$\begin{aligned}(\nabla W)_{ij}^{V4} &= \frac{2}{3}(\nabla W)_{ij}^{C} + \frac{1}{3}(\nabla W)_{ij}^{U} \\ (\nabla W)_{ji}^{V4} &= \frac{2}{3}(\nabla W)_{ji}^{C} + \frac{1}{3}(\nabla W)_{ij}^{D}\,.\end{aligned}$$

NUMERICAL DISSIPATION OF SIXTH-ORDER: V6-SCHEME. An even less dissipative scheme has been proposed.[14] It is a more complex linear combination of gradients using centered, upwind and nodal P_1-Galerkin gradients. The nodal P_1-Galerkin gradient of P_i is related to cell C_i and is computed by averaging the gradients of all the tetrahedra containing vertex P_i:

$$(\nabla W)_{P_i} = \frac{1}{4\,|C_i|}\sum_{K\in C_i}|K|(\nabla W)|_K\,.$$

A sixth-order dissipation scheme is then obtained by considering the following high-order gradient:

$$\begin{aligned}(\nabla W)_{ij}^{V6} &= (\nabla W)_{ij}^{V4} - \frac{1}{30}\left((\nabla W)_{ij}^{U} - 2(\nabla W)_{ij}^{C} + (\nabla W)_{ij}^{D}\right) - \frac{2}{15}\left((\nabla W)_{M_i} - 2(\nabla W)_{P_i} + (\nabla W)_{P_j}\right) \\ (\nabla W)_{ji}^{V6} &= (\nabla W)_{ji}^{V4} - \frac{1}{30}\left((\nabla W)_{ij}^{D} - 2(\nabla W)_{ij}^{C} + (\nabla W)_{ij}^{U}\right) - \frac{2}{15}\left((\nabla W)_{M_j} - 2(\nabla W)_{P_j} + (\nabla W)_{P_i}\right)\,,\end{aligned}$$

where $(\nabla W)_{M_{i,j}}$ is the gradient at the points $M_{i,j}$ intersection of the line defined by P_iP_j and upwind-downwind tetrahedra. These gradients are computed by linear interpolation of the nodal gradients of faces containing $M_{i,j}$.

DERVIEUX LIMITER. The previous MUSCL schemes are not monotone. Therefore, limiting functions must be coupled with the previous high-order gradient evaluations to guarantee the TVD property of the scheme. The gradient is substituted by a limited gradient denoted $(\nabla W)_{ij}^{lim}$. Here, we consider the three-entries limiter introduced by Dervieux which is a generalization of the Superbee limiter:[15]

$$\begin{aligned}&\text{if } uv \le 0 \text{ then} \\ &\quad Lim(u,v,w) = 0 \\ &\text{else} \\ &\quad Lim(u,v,w) = Sign(u)\,min(2\,|u|, 2\,|v|, |w|)\,,\end{aligned}$$

and we use: $Lim((\nabla W)_{ij}^{C}, (\nabla W)_{ij}^{D}, (\nabla W)_{ij}^{HO})$ where $(\nabla W)_{ij}^{HO}$ is even $(\nabla W)_{ij}^{V4}$ or $(\nabla W)_{ij}^{V6}$.

3. Discretization of the viscous terms

In `Wolf`, viscous terms are discretized with the finite element method (FEM). We evaluate viscous terms of the form:

$$\int_{\partial C_i}\mathbf{S}(W_i)\cdot\mathbf{n}\,\mathrm{d}\gamma = \sum_{P_j\in\mathcal{V}(P_i)}\int_{\partial C_{ij}}\mathbf{S}(W_i)\cdot\mathbf{n}\,\mathrm{d}\gamma + BT$$

where ∂C_{ij} is the common interface between cells C_i and C_j, and BT are boundary terms. Let ϕ_i the P_1 finite element basis function associated with vertex P_i, we have: $\int_K \nabla\phi_i\,\mathrm{d}\mathbf{x} = -\int_{\partial C_i\cap K}\mathbf{n}\,\mathrm{d}\gamma$ and if we assume that $\mathbf{S}(W_i)$, which comes from a gradient, is constant on element K, then we get:

$$\sum_{P_j\in\mathcal{V}(P_i)}\int_{\partial C_{ij}}\mathbf{S}(W_i)\cdot\mathbf{n}\,\mathrm{d}\gamma = -\sum_{K\ni P_i}\int_K\mathbf{S}(W_i)|_K\cdot\nabla\phi_i\,\mathrm{d}\mathbf{x}\,.$$

We notice that the finite element discretization is equivalent to the finite volume one. The effective computation of the previous integral then leads to the computation of integrals of the following form:

$$\int_K\nabla\phi_i\,\nabla\phi_j\,\mathrm{d}\mathbf{x} = |K|\,\nabla\phi_i|_K\,\nabla\phi_j|_K.$$

In this expression, $\nabla\phi_i|_K$ is the constant gradient of basis function ϕ_i associated with vertex P_i. This discretization is justified because the characteristic times associated with the diffusive terms are large as compared to the characteristic times associated with the hyperbolic (convective) terms. We now apply the FEM formulation to all convected variables that are averaged on the element and we easily verify that the components of the (Cauchy) stress tensor $\mathbf{S}(W)$ are constant on each element K. For instance, the term $u\,\tau_{xy}$ of the (Cauchy) stress tensor reads:

$$(u\,\tau_{xy})|_K = u|_K\,\mu|_K \sum_{\mathbf{p}_i\in K}\left(u_i\frac{\partial\phi_i|_K}{\partial y} + v_i\frac{\partial\phi_i|_K}{\partial x}\right).$$

The other terms are computed analogously.

The Spalart-Allmaras dissipation term is also discretized with the FEM:

$$\Phi^{SA}_{visc,K}(W_i,W_j,W_k,W_l) = |K|\,\frac{1}{\sigma}\rho_i\Big((\nu|_K+\tilde{\nu}|_K)\,\nabla\tilde{\nu}|_K\cdot\nabla\phi_i|_K\Big).$$

4. Discretization of the source terms

Finally, the Spalart-Allmaras source terms (diffusion, production and destruction) are discretized by simple integration on each vertex cell:

$$\Phi^{SA}_{source}(W_i) = |C_i|\left(\rho_i c_{b1}\tilde{S}_i\tilde{\nu}_i - \rho_i c_{w1} f_w\left(\frac{\tilde{\nu}_i}{d_i}\right)^2 + \frac{c_{b2}}{\sigma}\rho_i\|\nabla\tilde{\nu}_i\|\right),$$

where $|C_i|$ is the volume of the vertex cell and all variables are point-wise: $\tilde{S}_i = \Omega_i + \dfrac{\tilde{\nu}_i}{\kappa^2 d_i^2} f_{v2}$.

C. Boundary conditions

For presented flow simulations, three boundary conditions are required. Slip boundary conditions are imposed for bodies when the flow is considered inviscid or for symmetry. For viscous flow, no slip boundary conditions are considered for bodies. And finally, we used Steger-Warming flux to set up free-stream (external flow) conditions.

Slip condition For this boundary condition we impose weakly $\mathbf{u}\cdot\mathbf{n}=0$, which is done by imposing the following boundary flux:

$$\Phi_{slip}(W_i,T) = \sum_{T\ni P_i}\int_{\partial C_i\cap T}\mathbf{F}_{slip}(W_i)\cdot\mathbf{n}_T\,\mathrm{d}\gamma \quad\text{with}\quad \mathbf{F}_{slip}(W_i)\cdot\mathbf{n}_T = (0,\ p_i\,\mathbf{n}_T,\ 0)^t.$$

where T are boundary faces with normals $\mathbf{n}_T$. According to,[16] the slip boundary conditions for the turbulent equations is different if a wall or a symmetry plane is considered:

$$\tilde{\nu}_{slipwall} = 0 \quad\text{and}\quad \frac{\partial\tilde{\nu}_{symmetry}}{\partial\mathbf{n}} = 0.$$

No slip condition. Adiabatic conditions are considered, therefore only a null velocity is imposed strongly for this boundary condition: $\mathbf{u}=0$. The turbulent variable is also strongly imposed to zero: $\tilde{\nu}_{noslip}=0$.

Free-stream condition. This condition imposes a free-stream uniform flow from the infinite. It applies when we have a boundary Γ_∞ for which the infinite constant state W_∞ is prescribed:

$$W_\infty = (\rho_\infty,\ (\rho\mathbf{u})_\infty,\ (\rho E)_\infty)^t \quad\text{and}\quad \tilde{\nu}_{farfield}\in[3\nu_\infty\,,\,5\nu_\infty]\,.$$

This state enables upwind fluxes at the infinite to be computed. The considered boundary fluxes are built from a decomposition following the characteristics values. We consider the Steger-Warming flux which is completely upwind on solution W_i:

$$\Phi_\infty(W_i,T) = A^+(W_i,\mathbf{n_T})W_i + A^-(W_i,\mathbf{n_T})W_\infty \quad\text{where}\quad A^+ = \frac{|A|+A}{2} \quad\text{and}\quad A^- = \frac{|A|-A}{2}.$$

D. Time integration

1. Implicit scheme

For an explicit time discretization, the semi-discretized RANS system reads:

$$\frac{|C_i|}{\delta t_i^n}\left(W_i^{n+1}-W_i^n\right) \;=\; R_i(W^n)$$

where $R_i(W^n) = -\sum_{j\in\mathcal{V}(i)} \Phi_{hllc}(W_i^n, W_j^n, \mathbf{n}_{ij}) + \sum_{K\ni P_i} \Phi_{visc,K}(W_i^n, W_j^n, W_k^n, W_l^n) + \Phi_{source}(W_i^n) - \sum_{T\ni P_i} \Phi_{bc,T}(W_i^n, \mathbf{n}_T)$.
For implicit time discretization, we have :

$$\frac{|C_i|}{\delta t_i^n}\left(W_i^{n+1}-W_i^n\right) \;=\; R_i(W^{n+1})$$

which is linearized as:

$$\left(\frac{|C_i|}{\delta t_i^n} I_d - \frac{\partial R_i}{\partial W}(W^n)\right)\left(W_i^{n+1}-W_i^n\right) \;=\; R_i(W^n)$$

where $\frac{\partial R_i}{\partial W}(W^n)$ contributes the i^{th} line of the matrix and the following linearization have been done (with the notation $\delta W = W^{n+1} - W^n$):

$$\begin{aligned}
\Phi_{hllc}(W_i^{n+1}, W_j^{n+1}, \mathbf{n}_{ij}) &= \Phi_{hllc}(W_i^n, W_j^n, \mathbf{n}_{ij}) + \frac{\Phi_{hllc}}{\partial W_i}(W_i^n, W_j^n, \mathbf{n}_{ij})\delta W_i + \frac{\Phi_{hllc}}{\partial W_j}(W_i^n, W_j^n, \mathbf{n}_{ij})\delta W_j \\
\Phi_{visc,K}(W_i^{n+1}, W_j^{n+1}, W_k^{n+1}, W_l^{n+1}) &= \Phi_{visc,K}(W_i^n, W_j^n, W_k^n, W_l^n) \\
&+ \frac{\Phi_{visc,K}}{\partial W_i}(W_i^n, W_j^n, W_k^n, W_l^n)\delta W_i + \frac{\Phi_{visc,K}}{\partial W_j}(W_i^n, W_j^n, W_k^n, W_l^n)\delta W_j \\
&+ \frac{\Phi_{visc,K}}{\partial W_k}(W_i^n, W_j^n, W_k^n, W_l^n)\delta W_k + \frac{\Phi_{visc,K}}{\partial W_l}(W_i^n, W_j^n, W_k^n, W_l^n)\delta W_l \\
\Phi_{source}(W_i^{n+1}) &= \Phi_{source}(W_i^n) + \frac{\Phi_{source}}{\partial W_i}(W_i^n)\delta W_i \\
\Phi_{bc,T}(W_i^{n+1}, \mathbf{n}_T) &= \Phi_{bc,T}(W_i^n, \mathbf{n}_T) + \frac{\Phi_{bc,T}}{\partial W_i}(W_i^n, \mathbf{n}_T)\delta W_i \,.
\end{aligned}$$

Terms $\frac{\partial \Phi}{\partial W_i}$ and $\frac{\partial \Phi}{\partial W_i}$ contribute to the matrix diagonal block and extra-diagonal block, respectively, and terms Φ contributes to the right-hand side.

We then rewrite the linearized system in compact form:

$$\mathbf{A}^n\,\delta\mathbf{W}^n = \mathbf{R}^n$$

$$\text{where}\quad \mathbf{A}^n = \frac{|C|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{W}} \quad\text{and}\quad \delta\mathbf{W}^n = \mathbf{W}^{n+1} - \mathbf{W}^n\,.$$

2. Newton's method

To solve the non-linear system, we follow the approach based on Lower-Upper Symmetric Gauss-Seidel (LU-SGS) implicit solver initially introduced by Jameson[17] and fully developed by Sharov et al. and Luo et al.[18,19,20,21] The Newton's method can be either the LU-SGS approximate factorization or the SGS relaxation or the GMRES method with LUSGS or SGS as preconditioner. The LU-SGS and SGS are very attractive because they use an edge-based data structure which can be efficiently parallelized with p-threads.[22,21] From our experience, we have made the following - crucial - choices to solve the compressible Navier-Stokes equations.

Converging the Newton's method is important for the global convergence of the Navier-Stokes non-linear problem. Hence, an iterative method is required such as SGS or GMRES+LUSGS or GMRES+SGS[a]. Usually, the Newton's method iterates until the residual of the linear system is reduced by two orders of magnitude: 0.01.

[a] In comparison, the LU-SGS method works well for the compressible Euler equation

The choice of the renumbering also impacts strongly the convergence of the non-linear system. While Hilbert-type (space filling curve) renumbering is very efficient for cache misses and memory contention,[22,21] Breadth-first search renumbering proves to be more effective for the convergence of the implicit method and the overall efficiency.

Luo et al.[18,19,21] proposed to use a simplified flux function - a Rusanov approximate Riemann solver for the convective terms and the operator spectral radius for the viscous terms - to compute Jacobians while keeping the complex flux function for the right-hand side term. But, we observed that this modification slow down the convergence of the whole process. We found very advantageous to fully differentiate the HLLC approximate Riemann solver,[23] the FEM viscous terms and the Spalart-Allmaras source terms.[24]

To achieve high efficiency, automation and robustness in the resolution of the non-linear system of algebraic equations to steady-state, it is mandatory to have a clever strategy to specify the time step. This is done by coupling local under-relaxation coefficient and local CFL, see Section D 4.

In this work, we have considered the symmetric Gauss-Seidel (SGS) relaxation. This linear system can be re-written:

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^n = \mathbf{R}^n + (\mathbf{L}\mathbf{D}^{-1}\mathbf{U})\,\delta\mathbf{W}^n$$

The following approximate system is used:

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^n = \mathbf{R}^n\,.$$

Matrix $(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})$ can inverted in two sweeps which correspond to the LU-SGS approximate factorization:

$$\begin{array}{lrcl} \text{Forward sweep:} & (\mathbf{D}+\mathbf{L})\,\delta\mathbf{W}^* & = & \mathbf{R} \\ \text{Backward sweep:} & (\mathbf{D}+\mathbf{U})\,\delta\mathbf{W} & = & \mathbf{D}\delta\mathbf{W}^*\,. \end{array}$$

This sweeps can be written point wise:

$$\begin{aligned} \delta W_i^* &= D_{ii}^{-1}\Big(R_i - \sum_{j\in\mathcal{L}(i)} L_{ij}\delta W_j^*\Big) \\ \delta W_i &= \delta W_i^* - D_{ii}^{-1}\sum_{j\in\mathcal{U}(i)} U_{ij}\delta W_j^*\,. \end{aligned}$$

where $\mathcal{L}(i)$ (resp. $\mathcal{U}(i)$) is the set of vertices with an index lower (resp. upper) that i. The lower and upper parts can be stored or not (i.e., matrix-free) as choice between efficiency or memory requirements.

In the SGS relaxation, we first zero the unknown: $\delta W^0 = 0$. Then, k_{max} sub-iterations are made using forward and backward sweeps:

$$\begin{aligned} (\mathbf{D}+\mathbf{L})\,\delta\mathbf{W}^{k+1/2} &= \mathbf{R} - \mathbf{U}\,\mathbf{W}^k \\ (\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^{k+1} &= \mathbf{R} - \mathbf{L}\,\mathbf{W}^{k+1/2}\,. \end{aligned}$$

or rewritten point wise:

$$\begin{aligned} \delta W_i^{k+1/2} &= D_{ii}^{-1}\Big(R_i - \sum_{j\in\mathcal{L}(i)} L_{ij}\delta W_j^{k+1/2} - \sum_{j\in\mathcal{U}(i)} U_{ij}\delta W_j^{k}\Big) \\ \delta W_i^{k+1} &= D_{ii}^{-1}\Big(R_i - \sum_{j\in\mathcal{U}(i)} U_{ij}\delta W_j^{k+1} - \sum_{j\in\mathcal{L}(i)} L_{ij}\delta W_j^{k+1/2}\Big)\,. \end{aligned}$$

For one sub-iteration, the SGS method is equivalent to the LU-SGS method.

3. Time step computation

The local time step is computed at each vertex:

$$\delta t = \mathrm{CFL}\,\frac{h^2}{h\,(c+\|\mathbf{u}\|) + 2\,\dfrac{\gamma}{\rho}\left(\dfrac{\mu}{\mathrm{Pr}} + \dfrac{\mu_t}{\mathrm{Pr}_t}\right)}$$

where Pr= 0.72 and $\mathrm{Pr}_t = 0.9$ are the Prandtl and the turbulent Prandtl constants and, μ and μ_t are the (molecular dynamic) viscosity and the turbulent (molecular dynamic) viscosity.

4. CFL Law

Many CFL laws exist in the literature - linear, geometric, residual based, ... - but these laws generally require parameters that are difficult to establish optimally because they depend on the considered flow, the geometry and the size of the mesh. In other words, they dependent too much user's data. But, they are mandatory to achieve fast convergence in solving non-linear equations.

To avoid this issue, Luke et al. proposed a new approach[25] based on bounding the primitive variables, ρ, p and T, variations at each time step. More precisely, initially we allowed the maximal time step at each vertex, then this local time step is truncated such that the change in ρ, p and T are below a user given percentage η. But, the change in primitive variables during a given interval of time has to be estimated. A way to accomplish this is to solve an explicit time-integration step to describe a functional relationship between time and primitive variable. Notice that it is done before assembling the matrix and the truncated local time step is used to compute the mass matrix.

This method achieves a maximal efficiency as each vertex is progressing at its own optimal time step. But, that choice is made from an estimation before the linear system resolution, thus there is no guarantee on the convergence of the Newton's method.

Another attractive approach has been proposed by Burgess and Glasby[26] which couples under-relaxation coefficient and dynamic CFL. Here, the solution is analyzed at each step of the Newton's method (after solving the linear system) and before updating the solution. First, the change in primitive variables, ρ and p, is again controlled by a user given percentage η and defines a under-relaxation coefficient ω^n at each step of the process. This global coefficient is then applied to the solution evolution: $W^{n+1} = W^n + \omega^n \delta W$. Then, the CFL value is updated depending on that under-relaxation coefficient:

$$CFL^{n+1} = \begin{cases} 0.1\,CFL^n & \text{if} \quad \omega^n < 0.1 \\ CFL^n & \text{if} \quad 0.1 \leq \omega^n < 1 \\ \alpha CFL^n + \beta & \text{if} \quad \omega^n = 1 \end{cases}$$

where we choose $\alpha = 1$ and $\beta = 1$ for a linear increase or $\alpha = 2$ and $\beta = 0$ for a geometric increase. This adaptive CFL, thus time step, is attractive because it is based on the behavior of the Newton's method. To improve even more the robustness of the method, they propose to set the solution update to zero when the value of ω^n is less than 0.1, *i.e.,* $\omega^n = 0$.

This approach is extremely robust because if the Newton's method diverges, the current step is cancelled and the time step, via the CFL, is automatically reduced. But the considered criterium is global and hence one bad vertex in the mesh can kill the overall efficiency by not allowing the CFL to grow.

In `Wolf`, we consider an hybrid method having the efficiency of the first method and the robustness of the second one. We proceed exactly like the second approach but the under-relaxation coefficient is set locally, *i.e.,* vertex-wise, and each vertex is supplied with its own CFL coefficient which evolves with respect to its own under-relaxation coefficient. Thus, we have a local time step and a local CFL for each vertex.

E. Validation of the flow solver

In this Section, we compare `Wolf`, our in-house flow solver, to other codes for several Reynolds-Averaged Navier-Stokes (RANS) simulations. These comparisons ensure that the turbulence model has correctly been implemented in `Wolf`. This validation step is mandatory to confidently rely on the results of more complex simulations.

We compared `Wolf` to other codes for a large panel of verification and validation cases including flat plates, bumps, steps, airfoils... and we obtained relevant results. The material we used for comparison includes the test cases database released by NASA,[27] which provides flow conditions, grids and results from CFL3D[28] for comparison purpose.

We exhibit one verification case and two validation cases: a turbulent bump-in channel, a NACA 0012 airfoil and a transsonic RAE 2822 airfoil.

Bump-in-channel This test case is quite similar to a turbulent flat plate, except that the lower wall is curved. It was run at Mach $M = 0.2$ and at a Reynolds number of 3 Million based on length 1 of the mesh.

Speed isolines are depicted in Figure 1 and extracted velocity profiles in Figure 2. The results obtained with `Wolf` are similar to those from CFL3D and FUN3D.

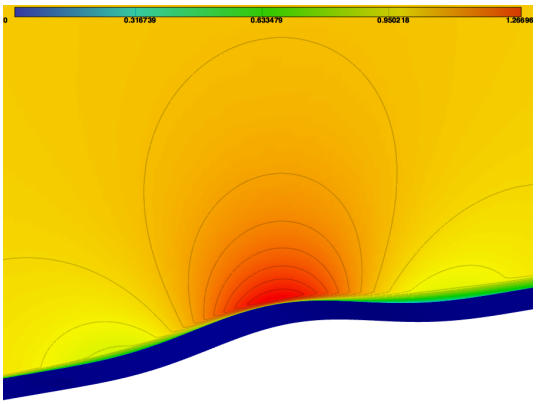

Figure 1. Speed isolines around the bump

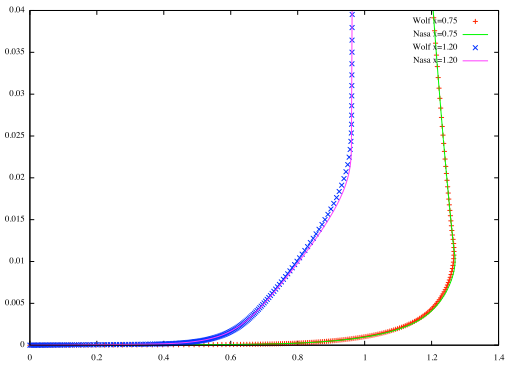

Figure 2. Bump: Velocity profiles

NACA 0012 Airfoil This test case is run at essentially incompressible conditions, the Mach number is $M = 0.15$. The Reynolds number is 6 Million based on airfoil chord $c = 1$. Pressure and speed isolines with an angle-of-attack $\alpha = 0$ are depicted in Figures 3 and 4. We compare pressure coefficients and skin friction coefficients for various angles-of-attack in Figure 5.

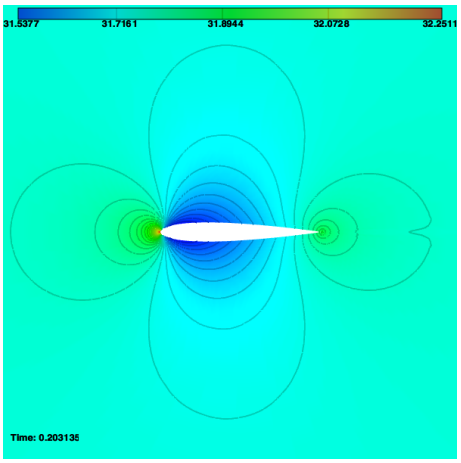

Figure 3. NACA0012: Pressure isolines ($\alpha = 0$).

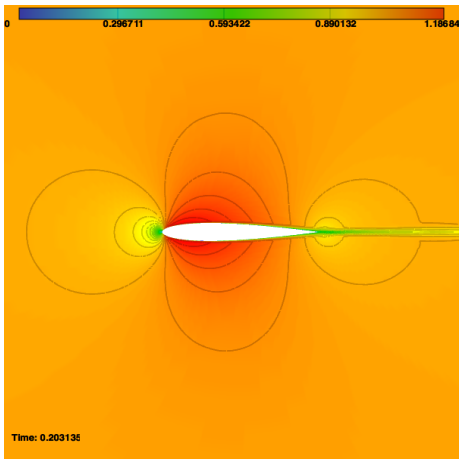

Figure 4. NACA0012: Speed isolines ($\alpha = 0$).

RAE 2822 Here we study a transsonic flow (Mach 0.7) around the RAE 2822 airfoil.[29] The Reynolds number is 6.5 Million based on airfoil chord $c = 1.5$ and the angle-of-attack is $\alpha = 2.31$ deg. Mach isolines are depicted in Figure 6. We compare[30] the pressure coefficients to experimental data and to Wind-US in Figure 7.

II. Meshing technologies

In this section, we review the main features of the adaptive mesh generator `feflo.a`. It is a metric-based mesh generator meaning that a metric tensor is used to prescribe the anisotropy in the domain. All the operators are based on local mesh modifications in order to guarantee that an adapted mesh is always provided on output. In the sequel, we describe respectively how the volume, surface, boundary layer and anisotropic cartesian mesh are generated.

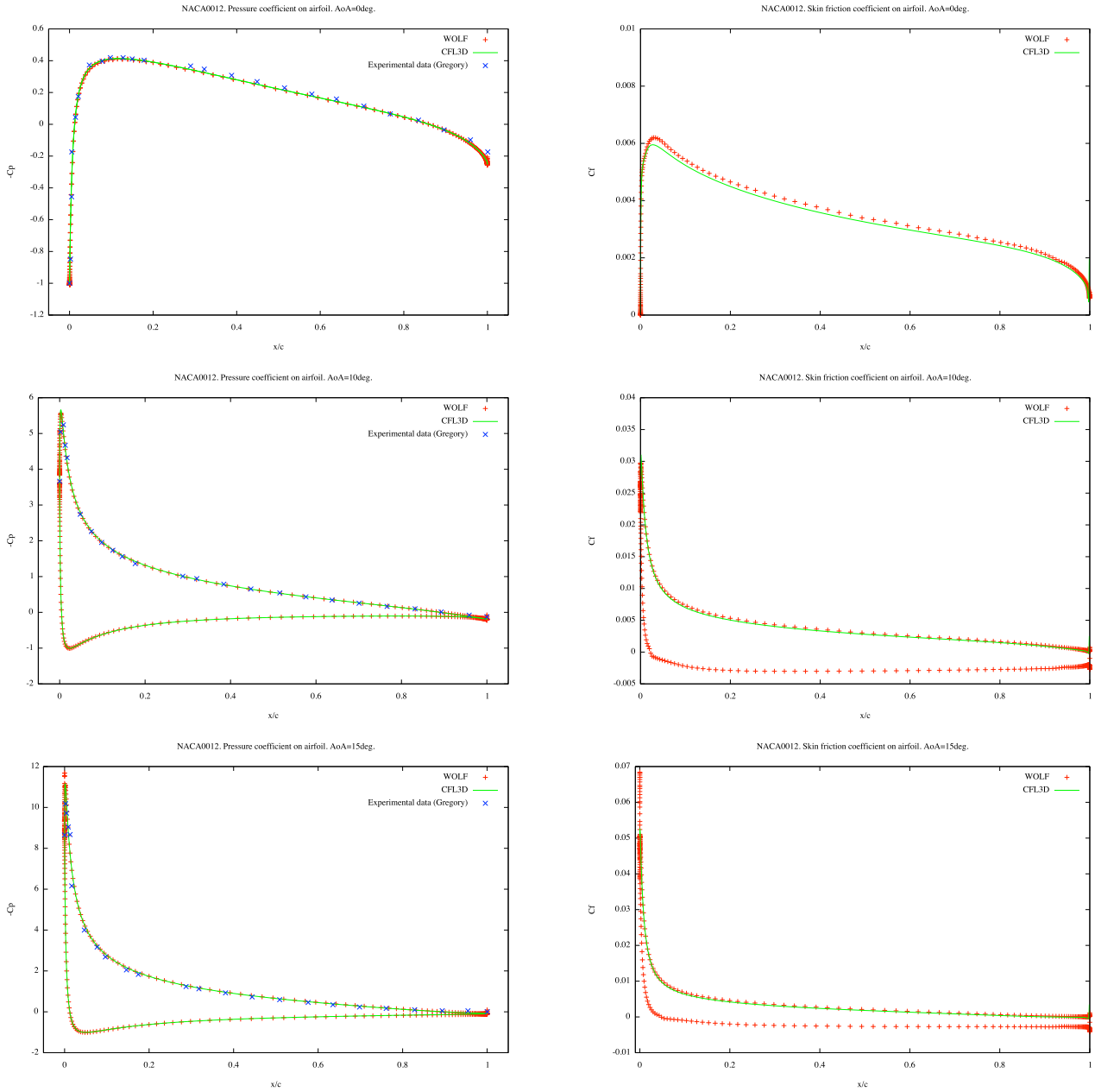

Figure 5. NACA0012: Pressure coefficient C_p and skin friction coefficient C_f for $\alpha = 0, 10, 15$ deg. Comparison with experimental data (C_p) and CFL3D (C_p and C_f).

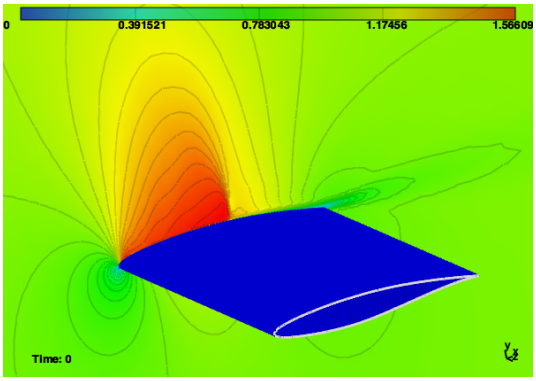

Figure 6. RAE 2822: Mach isolines.

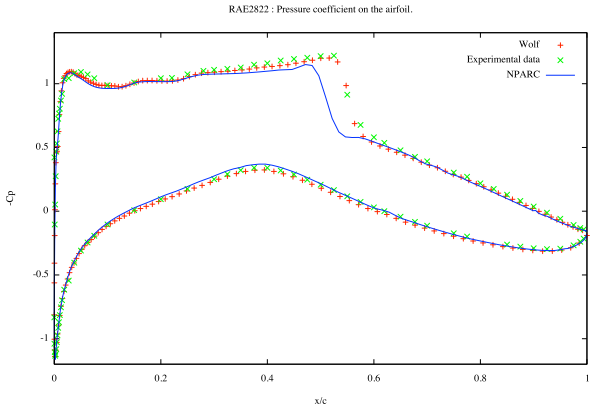

Figure 7. Comparison of the pressure coefficient around the RAE 2822 airfoil.

A. Volume and surface cavity-based operators

The mesh generator is based on the concept of unit mesh, which extends the notion of uniform mesh to anisotropic meshes. The anisotropy is prescribed through a metric field, so we suppose that a couple mesh and metric $(\mathcal{H}, \mathcal{M})$ is always provided. In this context, the adaptive mesh procedure aims at generating a unit mesh with respect to $\mathcal{M}$. A mesh is said to be unit when composed of almost unit-length edges. The length of an edge $\mathbf{e} = [A, B]$ in $\mathcal{M}$ is then evaluated with:

$$\ell_{\mathcal{M}}(A,B) \quad = \quad \int_0^1 \sqrt{{}^t AB\, \mathcal{M}((1-t)A + tB)\, AB}.$$

A first step in the adaptive mesh generation process thus consists in performing point insertions and edge collapses recursively until all edges have a length comprised in $[1/\sqrt{2}, \sqrt{2}]$. Then, an optimization procedure is added based on point smoothing, edges and faces swapping. During this step, we try to improve the overall quality of the mesh (in comparison to the unit mesh step where the convergence to a prescribed length distribution is the main concern). The quality is given in $\mathcal{M}$ by:

$$Q_{\mathcal{M}}(K) = \frac{36}{3^{\frac{1}{3}}} \frac{|K|_{\mathcal{M}}^{\frac{2}{3}}}{\sum_{i=1}^{6} \ell_{\mathcal{M}}^2(\mathbf{e}_i)} \in [0,1],$$

and a perfect unit element has a quality of 1.

In `feflo.a`, all the aforementioned mesh modification operators are based on an unique cavity-based operator revisited in metric-based context. This operator is inherited from Delaunay incremental insertion and can be written in a compact form as:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - \mathcal{C}_P + \mathcal{B}_P,$$

where $\mathcal{C}_P$ is the cavity of P and $\mathcal{B}_P$ the ball of P. In,[31] we derive several initializations and modifications of $\mathcal{C}_p$ to perform insertions, swaps, collapses and point smoothing. This operator also performs surface-based operations. In that case, a local surface approximation is computed based on a provided background surface mesh.

B. Hybrid cavity-based operators

The generation of a quasi-structured boundary layer mesh is based on the insertion of layers of hybrid entities (prisms when a triangulated surface is provided). The creation of hybrid entities is also based on a modified cavity-based operator. The modification consists in providing a list of initial faces from which the boundary layer must be extruded. This additional information is preserved and propagated into the domain for the subsequent layers. More precisely, starting from the initial surface mesh $\mathcal{S} = (F_i)_i$, a set of normals are

computed from $(F_i)_i$. Multi-normals are added in the case of singularities (an open ridge for example). We then list all the faces associated to a normal: $(\mathbf{n}_i, F_{k_1}, \ldots, F_{k_n})$. Then, the insertion of P is given by the following cavity operator:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - (\mathcal{C}_P - \mathcal{K}) + \mathcal{B}_P$$
$$\mathcal{S}_{k+1} = S_k - \cup_i F_{k_i} + \cup_i \tilde{F}_{k_i},$$

where $\mathcal{K}$ is the set of hybrid entities composing the previous or current layers and $\cup_i \tilde{F}_{k_i}$ are the new faces connected to P. This set of element defines a constraint on the cavity operator in order to avoid the removal of any elements defining the quasi-structured mesh.

In Figure 8, we represent the boundary layer mesh extrusion around a high lift geometry. In this example, 100 000 prisms/second are inserted on a Intel Core i7 at 2,7 Ghz. Note that no transition is applied, see Section B for detail smoothing and transition between the boundary layer and the remaining volume mesh.

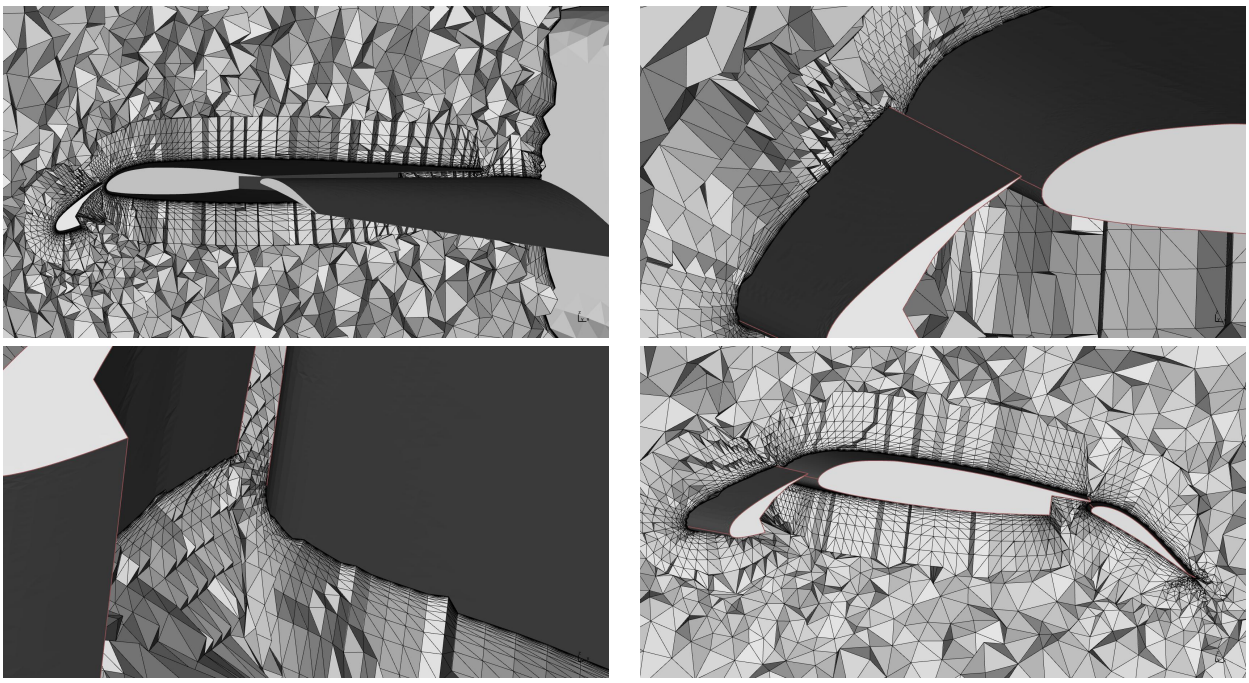

Figure 8. Boundary layer mesh generation around a high lift geometry.

C. Cartesian operators

Cartesian meshes are usually devised to generate quasi-structured meshes. In CFD, this meshes usually allows us to reduce transverse dissipation and to reduce the number of degrees of freedom by deactivating some edges/points when using appropriate finite volume cell definition (Barth cell for instance). However, the generation of unstructured cartesian meshes is usually combined with a frontal approach where a front is propagating into the domain while combining iteratively sequence of of (paving) tetrahedra in order to create regular structures. However, only uniform meshes are generally generated in 3D with this approach. In `feflo.a`, the cartesian operator relies on the previous cavity operator. The difference occurs in the insertion pattern. The new points are created along the eigenvector directions of the input metric. This choice allows us to favor orthogonally and alignment even with an anisotropic metric. When an isotropic metric is provided, alignment is along the natural axis.

III. Application to mesh adaptation for viscous flow simulations

Contrary to inviscid flow simulations, viscous flow simulations require a specific meshing of the boundary layers in the near wall regions. This is due notably to the dramatic variation in the normal direction of variables such as the velocity. The generation of quasi-structured meshes in these regions has proven to be

a reliable approach.[32,33,34,35,36,37,38,39,40,41,42]

Among the difficulties of boundary layer mesh generation include (i) having a smooth transition with the fully unstructured mesh used for the rest of the domain, (ii) its integration in the mesh adaptation loop described in the Introduction and (iii) the complex geometries often encountered in aeronautics.

In this section, we mainly focus on (ii) with a consideration for (i) and (iii). To this end, we combine the work introduced in Sections I and II to study how to couple anisotropic mesh adaptation with boundary layer mesh generation for complex geometries. Starting from a fully unstructured mesh adaptation scheme, improvements to generate and adapt quasi-structured boundary layer meshes are presented.

A. Fully unstructured mesh adaptation

In this first method, an initial semi-structured boundary layer mesh is mapped onto a metric tensor field. This metric field contains information on elements' orientation and size. Any standard mesh estimate is then used to generate a unit mesh with respect to the metric tensor field.

We apply this strategy to capturing shock/boundary layer interactions, see Figure 9. A supersonic flow (Mach 1.4) is applied across a double wedge wing. It generates a shock wave which interacts with a boundary layer at the bottom of the domain. We aim at capturing these interactions.

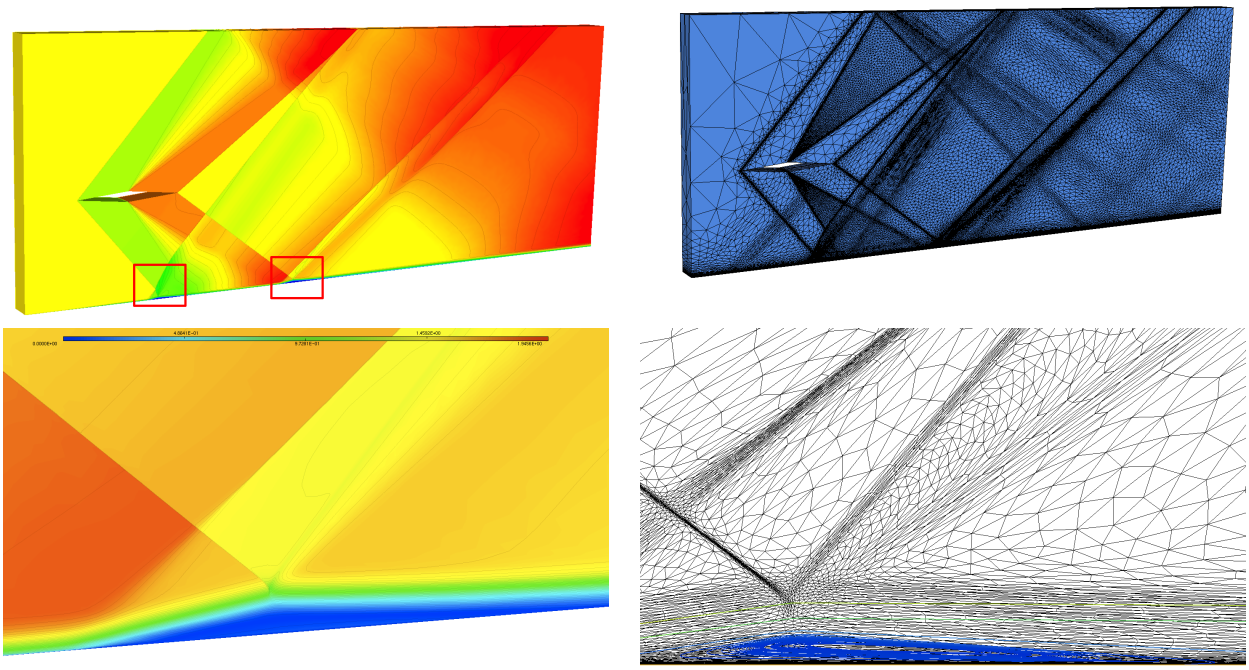

Figure 9. Shock/boundary layer interaction test case. Top, from left to right: Mach number isolines and final adapted mesh. Bottom: Velocity isolines in the interaction region and corresponding adapted mesh.

This simulation leads to the following observations:

- Generating a semi-structured boundary layer mesh extruded from the surface mesh gives only the require accuracy for the smaller layers. Indeed, the distance of the bottom of the shock from the viscous plate is around 10^{-3} whereas the initial height of the uniform boundary layer mesh was at 0.2. Consequently, it is not possible to keep a constant boundary layer mesh when an interaction occurs.
- This approach is completely generic and robust and can handle complex geometries. However, if the shock/boundary layer interaction is automatically handled, the impact of having a fully unstructured

mesh is not yet analyzed in term of solution accuracy and solver stability in the viscous area. Consequently, it seems also interesting to derive a method to generate structured mesh for the smaller layers (at least) while preserving (upper) anisotropic refinements.

This method has the advantage of simplifying the mesh adaptation mesh: one does not need a specific strategy for the near-wall regions. However, if the shock/boundary layer interaction is automatically handled, the impact of having a fully unstructured mesh is not yet analyzed in term of solution accuracy and solver stability in the viscous area. Consequently, it seems also interesting to derive a method to generate structured mesh for the smaller layers (at least) while preserving (upper) anisotropic refinements: this work is presented in B and C.

B. Unstructured mesh adaptation with fixed boundary layer mesh

It is not clearly established wether meshing near-wall regions with fully-unstructured meshes is suited for capturing viscous phenomena. Therefore, one might want to use semi-structured boundary layer meshes which have proven to be reliable. In this Subsection, we introduce a method[43] in which a boundary layer mesh is generated once at the beginning of the simulation. During the mesh adaptation loop, only the rest of the domain is adapted.

We describe the main steps of the adaptation process. First, the boundary layer mesh is generated according to a given wall spacing, a given number of layers and a given size gradation. Then, it is tagged and extracted: vertices on the interface between the two meshes are duplicated and new surface elements are created. We obtain two meshes: one boundary layer mesh and one we call domain mesh. The boundary layer mesh remains unchanged while the domain mesh is adapted. During the mesh adaptation of the domain, we make sure the vertices and triangles at the interface are not modified in order to be able to gather the two mesh parts.

We applied this method to a plume exhaust (see Figure 10) at Mach 2.2 with a Reynolds num- ber of 1.8 Million. The final mesh and mach isovalues are depicted in Figures 11 and 12. See Figure 13 for cuts in the volume.

To adapt the domain mesh, the solution we based the adaptation on is interpolated on the domain mesh and an hessian-based metric is computed. If there are several solutions of interest we want to adapt, we compute a metric for each one of them and perform a metric intersection. The metric field obtained for the domain will be used for a mesh adaptation.

A smooth transition between the boundary layer mesh and the rest of the domain is mandatory because it ensures a good behavior of the flow solver. In other words, we want no discontinuity in the elements' sizes. To do so, the natural geometric field of the boundary layer is propagated in the domain and intersected with the hessian-based metric.

This method has a major weakness: the skin of the plane is not adapted. Therefore it is not suited for capturing strong pressure variations on the wings for instance. Furthermore, this method requires too many mesh manipulations (splitting, gathering, metric gradation etc...). We want to devise a more simple approach with skin adaptation. Improvements are introduced in Subsection C.

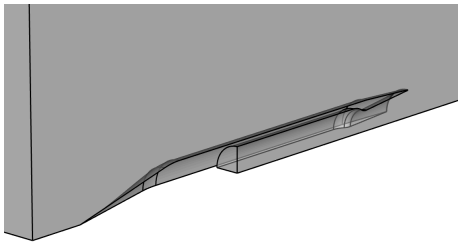

Figure 10. Initial domain of the plume exhaust.

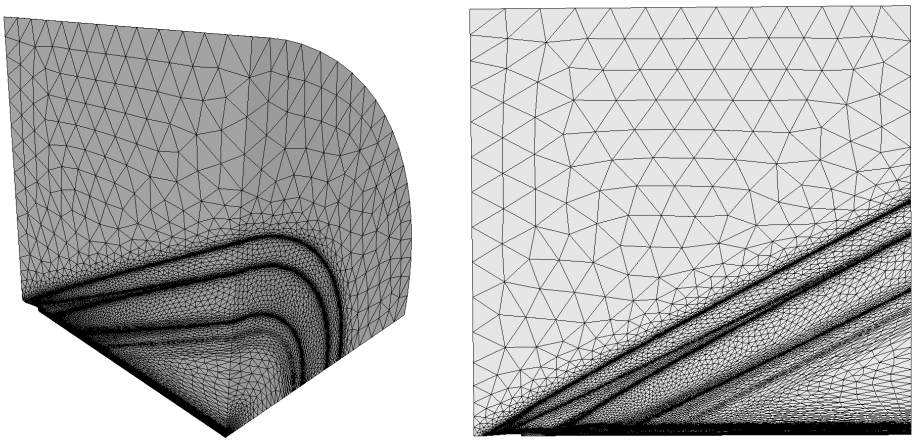

Figure 11. Final surface mesh of the domain.

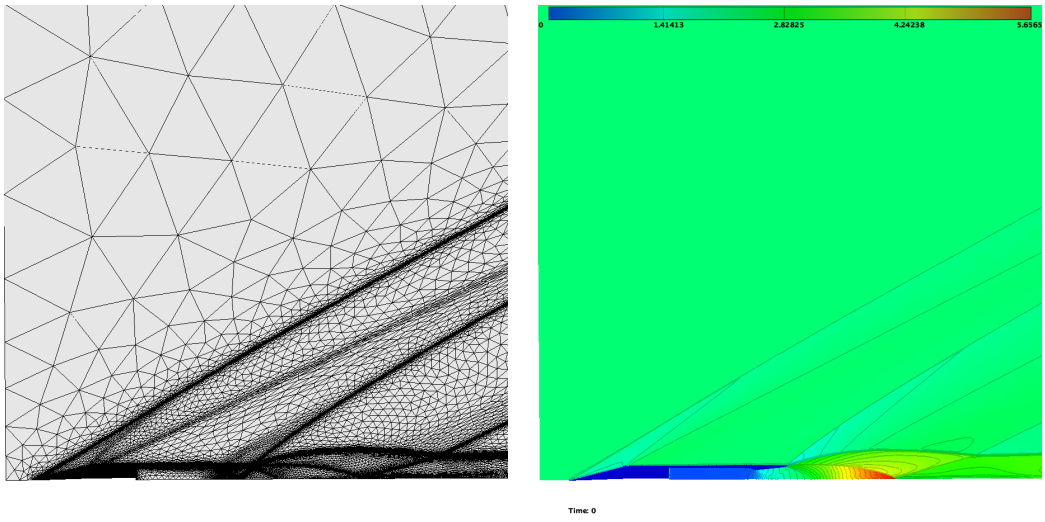

Figure 12. Surface mesh near the exhaust (left) and Mach iso-values (right).

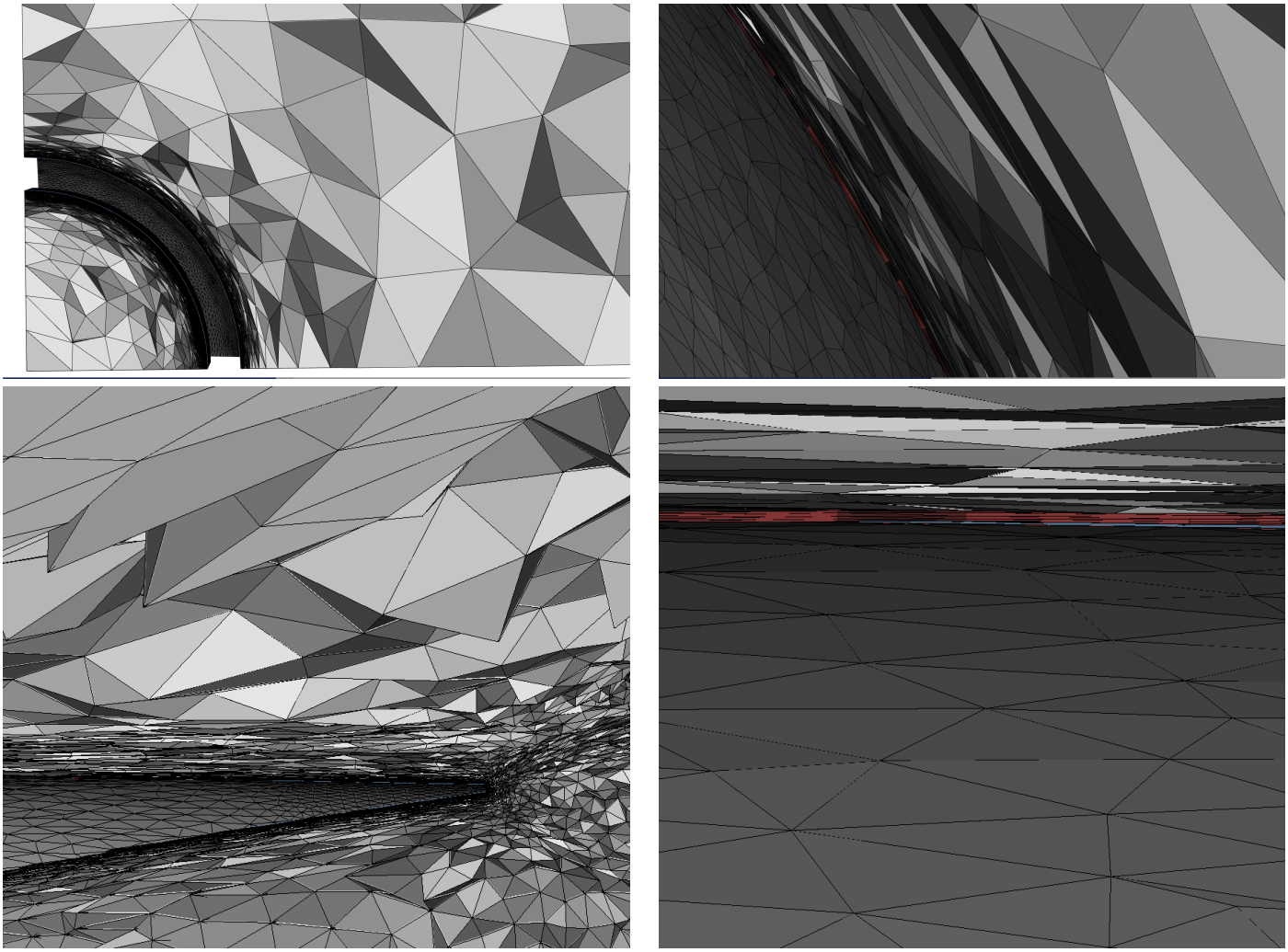

Figure 13. Cuts in the volume, red elements are part of the structured extruded layers

C. Hybrid mesh with skin adaptation

This method is quite similar to the one introduced in Subsection B, except that the boundary layer mesh is re-generated at each iteration in the mesh adaptation loop.
We proceed as follows. Given an initial fully unstructured mesh, a semi-structured boundary layer mesh is extruded. A flow solution and a metric tensor field are then computed. We then perform a fully-unstructured remeshing of the whole mesh according to that metric field.

The simulation leads to the following observations:

- This method allows to better capture anisotropic phenomena on the skin.
- It requires a strong robustness of the remesher. Re-generating a quasi-structured mesh from an anisotropic mesh presents challenging local configurations.

D. Cartesian approach

In the Cartesian approach, the boundary layer mesh is generated at the same time as the adapted volume mesh. To do so, a boundary layer metric is computed based on the distance to the viscous body. In the normal directions, the size is based on the desired size progression while in the tangential direction, the size is extrapolated from the surface mesh.

We illustrate the cartesian operator with an isotropic metric field around a f117 geometry at high angle of attack (20 degrees). This mesh is depicted in Figure 14 and is composed of 7 532 632 vertices and 4 572 1814 tetrahedra. It was generated in 7 min on an Intel Core i7 at 2.7 Ghz with 16 Gb of RAM. It is used to study the vortical flows generated by the delta wings. We expect to have a high level of details when observing the wake generated by the aircraft. From a flow solver point of view, this meshes enhances the high-order features of the solver (6th order for linear advection on structured meshes) by removing some high-order truncation error terms. Gradients of the solution are also better approximated.

If the previous example is isotropic, this approach works as well in an anisotropic context. We consider a transonic flow computation around a generic Falcon geometry at mach 0.8 with an angle of attack of 3 degrees. We control the interpolation error of the Mach number in $\mathbf{L}^2$ norms, the final mesh is obtained after 30 iteration and is composed of 1 110 735 vertices and 6 546 789, see Figure 15. The total cpu time is 40 minutes on an Intel Core i7 at 2.7Ghz.

IV. Conclusion and future work

The results of the code validation study led on the flow solver `Wolf` are satisfying. However, as a relatively young code several improvements remain to be achieved. We have recently implemented a full multigrid method and we are currently analyzing the results in order to increase the convergence gain. Performing mesh adaptations on every grid levels could be interesting. We intend to parallelize the solver in MPI in order to run simulations of cases of high complexity such as the high ligt prediction.[44]

We have experimented four different adaptive approaches for viscous simulations. The fully unstructured approach doesn't require any specific mesh nor metric manipulation (i.e. splitting, metric intersection, boundary layer extrusion etc.) for near-wall regions. That makes the adaptive process simple but it does not preserve the structure of the smaller layers. The structured mesh of the near-wall regions is preserved by the method presented in B and C, but at the cost of simplicity. We seek a completely generic method that preserves the structure. A first draw of this method is introduced inD. Based on the local cartesian operator presented in Section II, it

V. Acknowledgements

This work was partially funded by the Airbus Group Foundation.

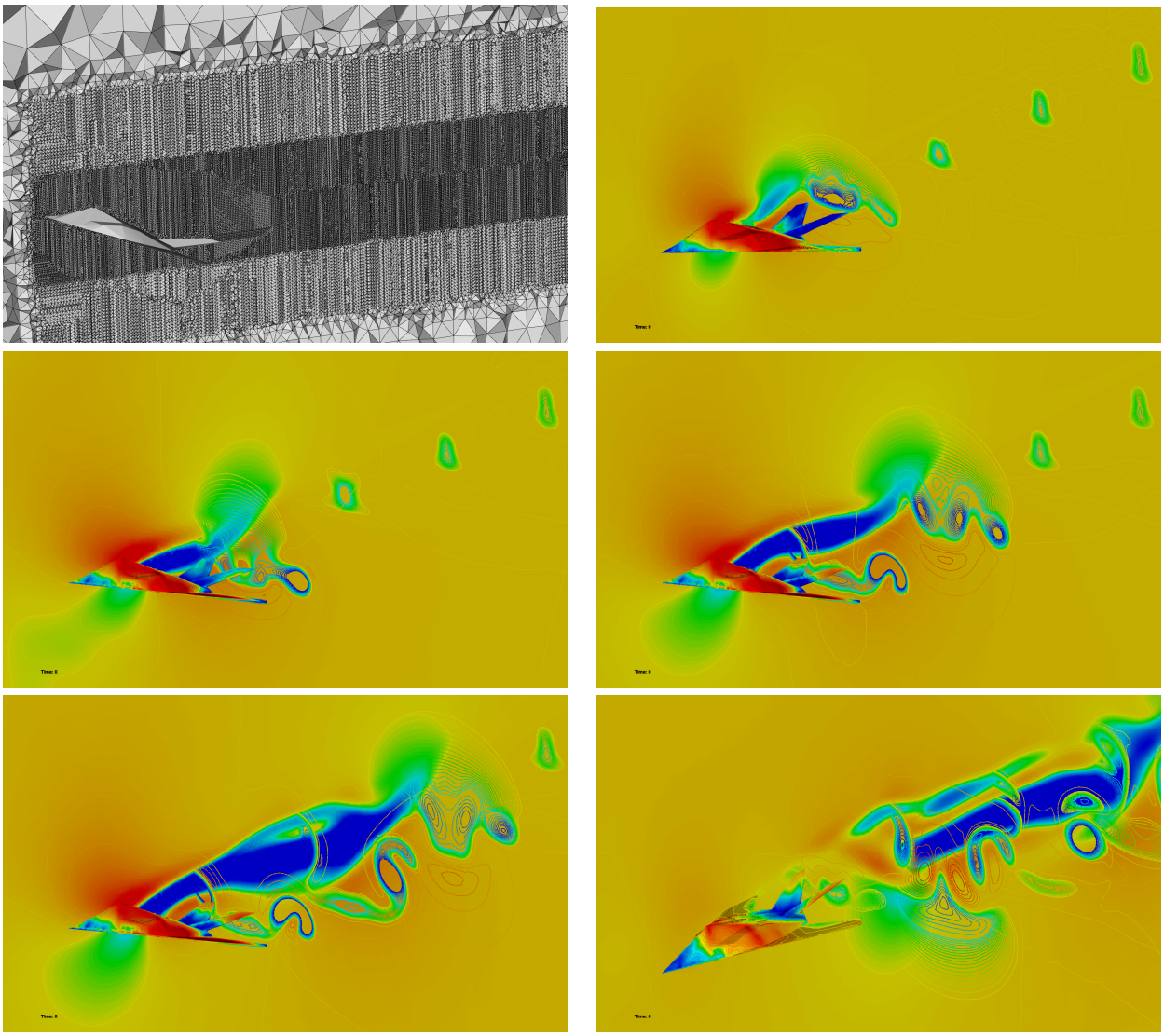

Figure 14. Cartesian mesh around a f117 geometry: Top left, cut in the volume mesh, then snapshots of the Mach number at different time steps showing the vortices and the wake of the aircraft.

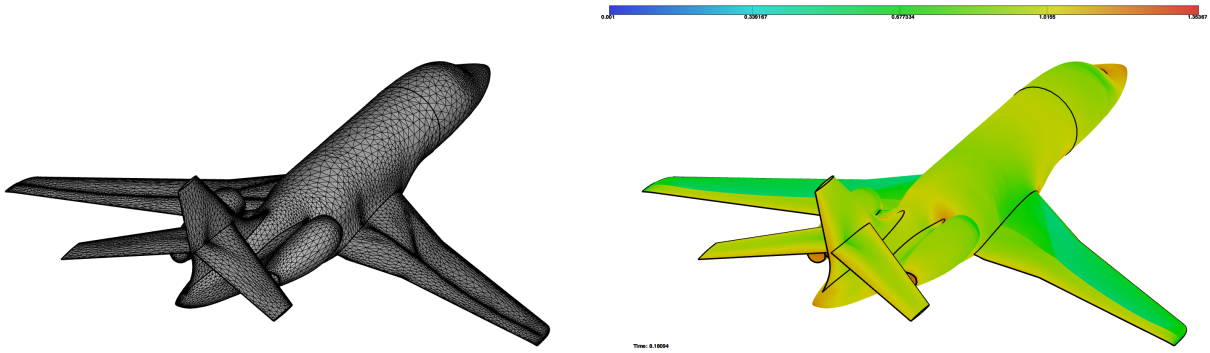

Figure 15. Final adapted surface mesh on the falcon geometry (left) and density iso-values (right).

References

[1]Castro-Díaz, M. J., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic Unstructured Mesh Adaptation for Flow Simulations," *Int. J. Numer. Meth. Fluids*, Vol. 25, 1997, pp. 475–491.

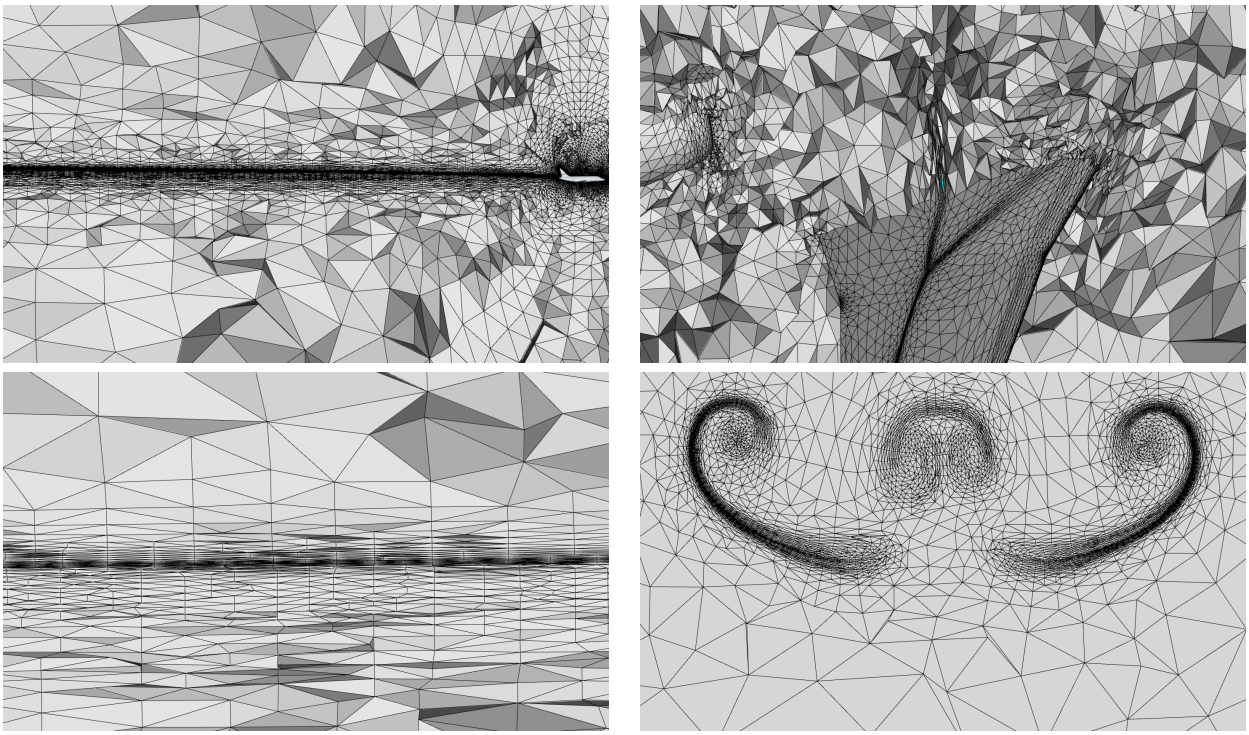

Figure 16. Different cuts in the final adapted volume mesh showing the quasi-aligned anisotropic refinements: in the wake (top and bottom left), near the lambda shock (top right) and 100, behind the falcon (bottom right).

[2]Frey, P. J. and Alauzet, F., "Anisotropic mesh adaptation for CFD computations," *Comput. Methods Appl. Mech. Engrg.*, Vol. 194, No. 48-49, 2005, pp. 5068–5082.

[3]Gruau, C. and Coupez, T., "3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric," *Comput. Methods Appl. Mech. Engrg.*, Vol. 194, No. 48-49, 2005, pp. 4951–4976.

[4]Li, X. L., Shephard, M. S., and Beall, M. W., "3D anisotropic mesh adaptation by mesh modification," *Comput. Methods Appl. Mech. Engrg.*, Vol. 194, No. 48-49, 2005, pp. 4915–4950.

[5]Pain, C. C., Umpleby, A. P., de Oliveira, C. R. E., and Goddard, A. J. H., "Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations," *Comput. Methods Appl. Mech. Engrg.*, Vol. 190, 2001, pp. 3771–3796.

[6]Tam, A., Ait-Ali-Yahia, D., Robichaud, M. P., Moore, M., Kozel, V., and Habashi, W. G., "Anisotropic mesh adaptation for 3D flows on structured and unstructured grids," *Comput. Methods Appl. Mech. Engrg.*, Vol. 189, 2000, pp. 1205–1230.

[7]Spalart, P. R. and Allmaras, S. R., "A one-equation turbulence model for aerodynamic flows," *AIAA-Paper 92-0439*, 1992.

[8]Eca, L., Hoekstra, M., Hay, A., and Pelletier, D., "Verification of RANS Solvers with Manufactured Solutions," *Eng. with Comput.*, Vol. 23, No. 4, Oct. 2007.

[9]Catris, S. and Aupoix, B., "Density corrections for turbulence models," *Aerospace Science and Technology*, Vol. 4, 2000, pp. 1–11.

[10]Barth, T., "Aspects of unstructured grids and finite-volume solver for the Euler and Navier-Stokes equations," *Von Karman Institute Lecture Series*, 1994.

[11]Gourvitch, N., Rogé, G., Abalakin, I., Dervieux, A., and Kozubskaya, T., "A tetrahedral-based super convergent scheme for aeroacoustics," RR-5212, INRIA, May 2004.

[12]Batten, P., Clarke, N., Lambert, C., and Causon, D. M., "On the choice of wavespeeds for the HLLC Riemann solver," *SIAM J. Sci. Comput.*, Vol. 18, No. 6, 1997, pp. 1553–1570.

[13]Leer, B. V., "Towards the ultimate conservative difference scheme I. The quest of monotonicity," *Lecture notes in physics*, Vol. 18, 1973, pp. 163–168.

[14]Debiez, C. and Dervieux, A., "Mixed-Element-Volume MUSCL methods with weak viscosity for steady and unsteady flow calculations," *Comput. & Fluids*, Vol. 29, 2000, pp. 89–118.

[15]Cournède, P.-H., Koobus, B., and Dervieux, A., "Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids," *European Journal of Computational Mechanics*, Vol. 15, No. 7-8, 2006, pp. 767–798.

[16]Allmaras, S., Johnson, F., and Spalart, P., "Modifications and clarifications for the implementation of the Spalart-Allamas turbulence model," *7th International Conference on Computational Fluid Dynamics*, Big Island, HI, USA, Jul 2012.

[17]Jameson, A. and Yoon, S., "Lower-Upper implicit schemes with multiple grids for the Euler equations," *AIAA Journal*, Vol. 25, No. 7, 1987, pp. 929–935.

[18]Luo, H., Baum, J., and Löhner, R., "A fast, matrix-free implicit method for compressible flows on unstructured grids," *J. Comp. Phys.*, Vol. 146, 1998, pp. 664–690.

[19]Luo, H., Baum, J., and Löhner, R., "An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids," *Comput. & Fluids*, Vol. 30, 2001, pp. 137–159.

[20]Sharov, D. and Nakahashi, K., "Reordering of hybrid unstructured grids for Lower-Upper Symmetric Gauss-Seidel computations," *AIAA Journal*, Vol. 36, No. 1, 1997, pp. 484–486.

[21]Sharov, D., Luo, H., Baum, J., and Löhner, R., "Implementation of unstructured grid GMRES+LU-SGS method on shared-memory, cache-based parallel computers," *AIAA Paper* , Vol. 2000-0927, 2000.

[22]Alauzet, F. and Loseille, A., "On the use of space filling curves for parallel anisotropic mesh adaptation," *Proceedings of the 18th International Meshing Roundtable*, Springer, 2009, pp. 337–357.

[23]Batten, P., Leschziner, M., and Goldberge, U., "Average-State Jacobians and Implicit Methods for Compressible Viscous and Turbulent Flows," *J. Comp. Phys.*, Vol. 137, 1997, pp. 38–78.

[24]Alauzet, F., "`Wolf` user guide. An edge-based Navier-Stokes flow solver based on the MEV numerical scheme," Internal report, INRIA, 2010.

[25]Luke, E., Tong, X.-L., Wu, J., Tang, L., and Cinnella, P., "A Step Towards Shape Shifting Algorithms: Reacting Flow Simulations Using Generalized Grids," *39th AIAA Aerospace Sciences Meeting*, AIAA Paper 2001-0897, Reno, NV, USA, Jan 2001.

[26]Burgess, N. and Glasby, R., "Advances in numerical methods for CREATE-AV analysis tools," *52th AIAA Aerospace Sciences Meeting*, AIAA Paper 2014-0417, National Harbor, MD, USA, Jan 2014.

[27]Rumsey, C., "Turbulence Modeling Resource," http://turbmodels.larc.nasa.gov/, Last updated 03/12/2013. Last checked 03/20/2013.

[28]Bananepos, S. K., Krist, S. L., Biedron, R. T., and Rumsey, C. L., "CFL3D User's Manual (Version 5.0)," 1998.

[29]Slater, J. W., "RAE 2822 Transonic Airfoil: Study 1," http://www.grc.nasa.gov/WWW/wind/valid/raetaf/raetaf01/raetaf01.html, Last updated 07/11/2008. Last checked 01/31/2013.

[30]Cook, P., Firmin, C., McDonald, A., and Establishment, R. A., *Aerofoil RAE 2822: Pressure Distributions, and Boundary Layer and Wake Measurements*, Technical memorandum / Royal Aircraft Establishment, RAE, 1977.

[31]Loseille, A. and Menier, V., "Serial and Parallel Mesh Modification Through A Unique Cavity-Based Primitive," *Proceedings of the 22nd International Meshing Roundtable*, edited by X. Jiao and J.-C. Weill, 2013.

[32]Aubry, R. and Löhner, R., "Generation of Viscous Grids With Ridges and Corners," *AIAA Paper* , Vol. 2007-3832, 2007.

[33]Bottasso, C. and Detomi, D., "A procedure for tetrahedral boundary layer mesh generation," *Engineering Computations*, Vol. 18, 2002, pp. 66–79.

[34]Garimella, R. and Shephard, M., "Boundary layer mesh generation for viscous flow simulations," *Int. J. Numer. Meth. Fluids*, Vol. 49, 2000, pp. 193–218.

[35]Hassan, O., Morgan, K., Probert, E. J., and Peraire, J., "Unstructured Tetrahedral Mesh Generation for Three-Dimensional Viscous Flows," *Int. J. Numer. Meth. Engng*, Vol. 39, No. 4, 1996, pp. 549–567.

[36]Ito, Y. and Nakahashi, K., "An approach to generate high quality unstructured hybrid meshes," *AIAA Paper* , Vol. 2006-0530, 2006.

[37]Löhner, R., "Generation of unstructured grids suitable for RANS calculations," *AIAA Paper* , Vol. 1999-0662, 1999.

[38]Marcum, D. L., "Adaptive Unstructured Grid Generation for Viscous Flow Applications," *AIAA Journal*, Vol. 34, No. 8, 1996, pp. 2440–2443.

[39]Pirzadeh, S., "Viscous unstructured three dimensional grids by the advancing-layers method," *AIAA Paper* , Vol. 1994-0417, 1994.

[40]Leicht, T. and Hartmann, R., "Error Estimation and Anisotropic Mesh Refinement for 3D Laminar Aerodynamic Flow Simulations," *J. Comput. Phys.*, 2010.

[41]Park, M. A. and Darmofal, D. L., "Parallel Anisotropic Tetrahedra Adaption," 2008.

[42]Chalot, F. L. and Perrier, P., *Industrial Aerodynamics*, 2004.

[43]Park, M. A. and Carlson, J. R., "Turbulent output-based anisotropic adaptation," *AIAA Paper* , Vol. 2011-0168, 2011.

[44]Rumsey, C., "High Lift Prediction Workshop," http://hiliftpw.larc.nasa.gov/Workshop2/, Last updated 05/14/2014. Last checked 05/20/2014.